REMARKS

Applicants respectfully request that the above-identified application be reexamined.

The final Office Action mailed on November 30, 2004 ("Office Action"), rejected all the claims in the application. More specifically, Claims 1 and 3-17 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Foody et al., U.S. Patent No. 5,732,270 (hereinafter "Foody"), in view of Katchabaw et al., "Making Distributed Applications Manageable Through Instrumentation," *The Journal of Systems and Software 45*(2):81-97, March 1, 1999 (hereinafter "Katchabaw"). The Office Action rejected Claim 2 under 35 U.S.C. § 103(a) as being unpatentable over Foody and Katchabaw, in view of Festor et al., "Integration of WBEM-Based Management Agents in the OSI Framework," Integrated Network Management, 1999, *Proceedings of the 6th IFIP/IEEE International Symposium* (hereinafter "Festor").

After carefully considering the Office Action, applicants modified the claims to further distinguish the claimed invention from the cited references. As amended, the claims now recite providing access to instrumentation data from within a managed code runtime environment, using an instrumentation client API within the runtime environment. Applicants respectfully submit that, as amended, the application is now clearly allowable in view of the cited and applied references.

Prior to discussing in detail why applicants believe that all of the claims in the application are allowable over the applied references, brief descriptions of applicants' invention and the cited references are provided. The following discussions of the disclosed embodiments of applicants' invention and the teachings of the applied references are not provided to define the scope or interpretation of any of applicants' claims. Instead, such discussed differences are provided to help the U.S. Patent and Trademark Office better appreciate important claim distinctions discussed thereafter.

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

Summary of the Invention

In general, the invention enables a managed code runtime environment to access information outside in a manner that is consistent with the classes and models provided by the runtime environment. More specifically, the invention provides an instrumentation client API within the runtime environment for providing access to outside instrumentation data. The instrumentation client API provides access to instrumentation data in a manner that is consistent with the classes and models provided by the runtime environment. The instrumentation client API also raises exceptions in a manner that is compatible with the runtime environment.

The instrumentation client API that is exposed within a managed code runtime environment wraps, or translates, calls to and from an instrumentation data source external to the runtime environment. Specifically, the invention provides an API within a managed code runtime environment that exposes a number of classes for accessing instrumentation data that resides within or outside the runtime environment. For instance, the invention provides a management object class within the runtime environment for representing instances of instrumentation data. An instance of the management object class encapsulates a single non-transient instrumentation data object.

Three pieces of information are necessary to construct an instance of the management object class: the path of the instrumentation data object to bind to, options used to retrieve the instrumentation data object, and a scope identifying the parent of the instrumentation data object. Once this information has been provided, a Get() method may be coded to bind an instance of the management object class to the corresponding instrumentation data object. If the operation is unsuccessful, an instance of a management exception class will be returned. The management exception class throws exceptions compatible with the runtime environment based upon error conditions returned from the instrumentation data object.

Once an instance of the management object class has been successfully constructed, the object provides access to the methods, qualifiers, and properties of the object in a manner that is

easy to use and consistent with the runtime environment. For instance, the management object encapsulates a number of methods that may be performed directly on the object itself, such as a Get() method for binding the instance of the management object class to the instrumentation data object, a Put() method for saving changes made to the object or creating a new instrumentation data object, a CopyTo() method for copying the object to another scope, and a Delete() method for deleting the object. Moreover, an InvokeMethod() method is encapsulated for invoking a method provided by the instrumentation data object, directly upon the object.

According to another aspect of the invention, the management object class also encapsulates methods for retrieving related objects that may be called directly on a management object. For instance, a GetRelated() method is provided that offers functionality for retrieving a collection of objects related to the instance of the management object class upon which it is called. A GetRelationships() method is also provided that offers functionality for retrieving a collection of objects that refer to the instance of the management object class upon which it is called. The collections of objects returned by these methods are compatible with the data types utilized in the runtime environment.

According to another aspect of the invention, an indexer is used to allow easy access to the properties of an instance of the management object class. Using this indexer, properties of a management object may be retrieved from the object itself in an array-like fashion. Direct retrieval of the properties of a management object in this way is consistent with the object-oriented programming paradigm of the managed code runtime environment and eliminates the need to call a method to retrieve properties of a management object.

According to yet another aspect of the invention, a management object searcher class is also provided to permit the retrieval of a collection of instrumentation data objects based on a specified query. A management options object may also be utilized to specify options for the search. A management event watcher class is further provided that incorporates functionality for subscribing to temporary event notifications from the management instrumentation data source.

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

A management operation watcher class is also provided for raising events concerning operations on other classes. Each of these classes are implemented in a manner that is consistent with the runtime environment in which they execute, allow access to methods, properties, and qualifiers in a similarly consistent manner, and throw exceptions in a manner that is also consistent with the runtime environment.

In summary, the invention provides a method and a system for providing applications executing within a managed code runtime environment easy access to instrumentation data that resides either inside or outside the runtime environment. The invention further ensures that access to such instrumentation data is in a manner that is consistent with the models and classes provided by the runtime environment.

Summary of Foody

Foody purportedly teaches a system and a method that enables objects from one or more heterogeneous object systems in a digital computer to interoperate bi-directionally and be combined in the creation of a larger object-oriented software project. In Foody, objects from a foreign object system are not modified, but appear to be native to the object system in which they are used or accessed. Foody accomplishes this function by using a native proxy object that is not distinguishable from other native objects in the object system in which a foreign object is used or accessed. This native proxy object is constructed for the foreign object. The native proxy object contains an identifier to the foreign object, as well as a pointer to a software description of how to access and manipulate the foreign object, i.e., how to call its methods, set its properties, and handle exceptions. Once the native proxy object is manipulated, it follows the instructions in the software description which, in turn, results in the corresponding manipulation of the foreign object.

In summary, Foody teaches creating a native proxy object in the object system in which the foreign object is used or accessed. Once a foreign object is requested, the native proxy object is manipulated instead. Nowhere does Foody teach an instrumentation client API within a

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

managed code runtime environment for accessing instrumentation data available outside this runtime environment.

Summary of Katchabaw

Katchabaw focuses on how distributed application processes can be made manageable. Katchabaw purportedly teaches instrumenting the distributed application processes to allow them to respond to management requests, generate management reports, and maintain information required by the management system. Katchabaw achieves this purpose by using an instrumentation architecture.

Katchabaw teaches instrumentation to be a process of inserting code into the application at strategic locations so a managed process can maintain management information, respond to information requests, and generate event reports. Using the instrumentation architecture, Katchabaw aims to make efforts toward automating some parts of an instrumentation process and providing guidance to facilitate the development of customer instrumentation in a controlled and structured manner.

Katchabaw teaches an instrumentation architecture containing three types of components: managers, which make decisions based on collected management information guided by management policies; management agents, which collect management information; and the managed objects, which represent actual system or network resources being managed.

A management agent is responsible for a particular set of managed objects. On one hand, the management agent receives management requests from managers and carries out the operations on the appropriate managed objects. On the other hand, the management agents route notifications emitted by manager objects to the management applications.

In essence, Katchabaw uses an agent between the manager application and the instrumentation data source to pass management operations from a manager application to an instrumentation data source or to pass event reports from the instrumentation data source to the manager application. Therefore, nowhere does Katchabaw teach an instrumentation client API

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

within a managed code runtime environment for accessing instrumentation data outside the runtime environment.

Summary of Festor

Festor purportedly teaches a set of mappings and an implementation of an integration agent allowing Web-based enterprise management agents to be managed by OSI-based management platform and applications. Expanding existing integration approaches, Festor provides three original items: the support of the CIM meta-model in an OSI agent, the mapping of relationships onto GRM specifications, and a full Java-based implementation of an integration agent, Q.Adapter. Specifically, Festor teaches two Java-based communication interfaces for the integration agent, one offering a client (manager) role API from the manager application side and one offering an agent-side API.

In summary, Festor provides an integration agent that has two communication interfaces. Nowhere does Festor teach an instrumentation client API that is within a managed code runtime environment. The present invention allows applications within a managed code runtime environment to access instrumentation data components outside the runtime environment through the instrumentation client API, instead of through an integration agent.

The Claims Distinguished

The Office Action rejected Claims 1 and 3-17 under 35 U.S.C. § 103(a) as being unpatentable over Foody in view of Katchabaw.

Independent Claim 1

In its amended form, Claim 1 reads as follows:

> 1. A method for providing access to instrumentation data from within a managed code runtime environment, comprising:
> providing an instrumentation client API within said runtime environment;
> receiving a request for instrumentation data available outside said runtime environment;

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

transmitting a request for said instrumentation data to an instrumentation data source external to said runtime environment, using the instrumentation client API;

receiving a response to said request to said instrumentation data source;

converting said response to a format that is compatible with said runtime environment; and

responding to said request for instrumentation data with said converted response.

The Office Action alleges that Foody and Katchabaw combined discloses each and every element of Claim 1 because Foody's native proxy object and Katchabaw's go-between agent are considered to be a data access interface in a runtime environment. See Office Action, page 10, first paragraph. Applicants disagree with such a broad interpretation of data access interface. However, to further progress the prosecution of this application, applicants amend Claim 1, which now recites providing access to instrumentation data from within a managed code runtime environment using an instrumentation client API. Applicants submit that Foody and Katchabaw, either alone or combined, do not teach the limitations recited by amended Claim 1.

For example, Foody does not teach providing an instrumentation client API within a managed code runtime environment. The present application describes an exemplary instrumentation client API. The exemplary API defines a number of classes for facilitating communication with an instrumentation data source. These classes allow a managed code application executing within a managed code environment to request, receive, and modify instrumentation data from the instrumentation data source. In addition, these classes provide access to instrumentation data from within the managed code environment in a manner that is consistent with the models and classes provided by the managed code runtime environment. See Detailed Description of the Invention at page 17, line 20, to page 19, line 20. As known by those skilled in the art, in a managed code runtime environment, code is executed by a common language runtime environment rather than directly by the operating system. The common language runtime environment provides managed code application services such as automatic

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

garbage collection, runtime type checking and security support. On the other hand, for applications that are not in a managed code runtime environment, code is executed by an operating system that is outside the common language runtime environment. Unmanaged code provides its own garbage collection, runtime type checking, security support, etc. Therefore, a managed code runtime environment is different from a native code runtime environment. Foody does not teach providing access to instrumentation data from within a managed code runtime environment. Rather, Foody teaches using a native proxy object in a system for a foreign object outside the system. Foody specifically teaches that the native proxy object contains an identifier to the foreign object, as well as a pointer to a software description of how to access and manipulate the foreign object, i.e., how to call its methods, set its properties, and handle exceptions. When the native proxy object is manipulated, it follows the instruction in the software description which, in turn, results in the corresponding manipulation of the foreign object. See Foody, Col. 6, line 60, through Col. 7, line 2. A native proxy object for a foreign object outside a system is not equivalent to an instrumentation client API in a managed code environment. Therefore, nowhere does Foody teach an instrumentation client API within a managed code runtime environment.

The Office Action correctly concludes that Foody does not teach instrumentation data. See Office Action, page 3. The Office Action alleges that Katchabaw makes up this deficiency by providing instrumentation components via an agent for data access. Applicants respectfully disagree.

First, there is no teaching or suggestion in Foody and Katchabaw, taken alone or in combination, why it would be obvious to combine the individual teachings of these references. More importantly, as noted above, even if these references were combinable--which applicants categorically deny--the resulting combination would not anticipate the subject matter of Claim 1 because neither Foody nor Katchabaw teaches providing an instrumentation client API for instrumentation data outside a managed code runtime environment. As the discussion above

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

shows, Foody does not teach instrumentation client API. Neither does Katchabaw. Katchabaw teaches three major components: the manager application, which reads information or event reports from the instrumentation data sources; the instrumentation data source, which provides event reports to the manager application; and an agent, which acts as a go-between to pass management operations from the manager application to the instrumentation data source, and to pass notifications from the instrumentation data source as event reports to the manager application. Nowhere does Katchabaw teach that the agent component is an instrumentation client API. Even if the agent component is an instrumentation client API--which applicants categorically deny--nowhere does Katchabaw teach that the agent component is in the same runtime environment as the manager application. As a result, neither Foody nor Katchabaw, taken alone or in combination, teaches the subject matter recited by Claim 1. Consequently, applicants respectfully submit that Claim 1 is clearly allowable.

Because Claims 3-6 depend from Claim 1, these claims are submitted to be allowable for at least the same reasons that Claim 1 is allowable. Claim 7 recites a computer-readable medium comprising instructions, which, when executed by a computer, cause the computer to perform the method of any one of Claims 1 and 3-6. Therefore, Claim 7 is allowable since, as noted above, Claims 1 and 3-6 are allowable. Claim 8 recites a computer-controlled apparatus capable of performing the method of any one of Claims 1 and 3-6. As the above discussion shows, Claims 1 and 3-6 are allowable.. Therefore, Claim 8 is also allowable.

Independent Claim 9

In its amended form, Claim 9 recites:

> 9. A method for accessing instrumentation data from within a managed code runtime environment, comprising:
>
> receiving a request to construct a management object comprising said instrumentation data;
>
> in response to said request, querying for said instrumentation data, using an instrumentation client API within said runtime environment;

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

determining whether said instrumentation data was successfully returned; and

in response to determining that said instrumentation data was successfully returned, constructing said management object and populating said management object with said instrumentation data.

Neither Foody nor Katchabaw teaches the subject matter recited by Claim 9. For example, Claim 9 specifically recites querying an instrumentation client API within the managed code runtime environment for instrumentation data external to this runtime environment. Nowhere does Foody teach such a limitation. As noted above, Foody teaches creating a native proxy object within the object system in which the foreign object represented by the native proxy object is used or accessed. As the discussion for Claim 1 shows, a native proxy object within an object system is not the same as an instrumentation client API with a managed code runtime environment.

The Office Action correctly concludes that Foody fails to disclose that the software components requested from the runtime environment are instrumentation data. The Office Action asserts that Katchabaw makes up this deficiency. As the discussion above relating to Claim 1 shows, even if Katchabaw does teach instrumentation data, Foody and Katchabaw combined still do not address all the limitations recited by Claim 9 since Katchabaw does not teach an instrumentation client API within a managed code runtime environment either.

Claims 10-15 depend from Claim 9. Therefore, they are allowable for the reasons that Claim 9 is allowable. Claim 16 is a computer-readable medium comprising instructions to perform the method of any one of Claims 9-15. Claim 17 is a computer-controlled apparatus capable of performing the method of any one of Claims 9-15. Therefore, Claims 16 and 17 are allowable for the same reasons that Claims 9-15 are allowable.

CONCLUSION

In view of the foregoing comments, applicants respectfully submit that all of the claims in this application are clearly allowable in view of the cited and applied references.

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

Consequently, early and favorable action allowing these claims and passing this application to issue is respectfully solicited. If the Examiner has any questions or comments concerning this application, the Examiner is invited to contact the applicants' undersigned attorney at the number below.

Respectfully submitted,

CHRISTENSEN O'CONNOR
JOHNSON KINDNESS PLLC

Joy Y. Xiang
Registration No. 55,747
Direct Dial No. 206.695.1607

I hereby certify that this correspondence is being transmitted via facsimile to the U.S. Patent and Trademark Office, Group Art Unit 2124, Examiner T.A. Vu at facsimile number 703.872-9306 on February 28, 2005.

Date: February 28, 2005 Patricia Gubble

JYX:ejh/nfs

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100